The Iterative Protein Redesign and Optimization (IPRO) Suite of Programs

Robert J. Pantazes,[a] Matthew J. Grisewood,[b] Tong Li,[b] Nathanael P. Gifford,[b] and Costas D. Maranas*[b]

Proteins are an important class of biomolecules with applications spanning across biotechnology and medicine. In many cases, native proteins must be redesigned to improve various performance metrics by changing their amino acid sequences. Algorithms can help sharpen protein library design by focusing the library on sequences that optimize computationally accessible proxies. The Iterative Protein Redesign and Optimization (IPRO) suite of programs offers an integrated environment for (1) altering protein binding affinity and specificity, (2) grafting a binding pocket into an existing protein scaffold, (3) predicting an antibody's tertiary structure based on its sequence, (4) enhancing enzymatic activity, and (5) assessing the structure and binding energetics for a specific mutant. This manuscript provides an overview of the methods involved in IPRO, input language terminology, algorithmic details, software implementation specifics and application highlights. IPRO can be downloaded at http://maranas.che.psu.edu. © 2014 Wiley Periodicals, Inc.

DOI: 10.1002/jcc.23796

Introduction

Proteins are amino acid polymers used for widespread biological functions including structural integrity (e.g., actin), catalysis of chemical reactions (e.g., cytochrome p450), and targeting of foreign molecules (e.g., antibodies). They are also used in diverse applications spanning biocatalysis in biofuel production[1,2] to receptor blocking in biomedicine.[3] Many protein engineering methods[4] have been devised to improve on native protein properties. Computational protein engineering methods[5] can reduce both time and cost by quickly ranking *in silico* binding affinity, thus streamlining the process of identifying improving mutations.

Computational protein design techniques can be divided into two categories: those that exclusively use sequence information to predict optimal recombination locations for combinatorial libraries[6–12] and those that use detailed, atomic-level information to predict favorable redesigns.[13–24] Generally, using more molecular detail leads to higher accuracy at the expense of longer computational times.[25] A number of successes at redesigning proteins to bind new targets,[26–28] improve stability,[29,30] and introduce novel enzymatic activity[17,31,32] allude to the promise of using computations for guiding protein design. For detailed reviews of computational protein design, we refer the reader to the following resources.[5,33–35]

The Iterative Protein Redesign and Optimization (IPRO)[36,37] method uses alternating protein backbone perturbations and amino acid sequence mutations to improve the binding of proteins to novel ligands. A number of computational tools that make use of the basic IPRO workflow have been customized for a variety of applications. For example, IPRO was applied to change the cofactor specificity of *Candida boidinii* xylose reductase from its native cofactor of nicotinamide adenine dinucleotide phosphate to nicotinamide adenine dinucleotide.[38] OptGraft[39] identifies the region in a protein that can best support a novel binding site and then finds mutations in neighboring residues to maintain the proper geometry of the binding pocket. OptGraft was applied to introduce a calcium-binding site from thioredoxin into the first domain of CD2. OptZyme[40] uses ground state substrates and transition state analogues to improve the catalytic properties of enzymes. The optimal complementarity determining regions (OptCDR)[41] method designs novel complementarity determining regions (CDRs), the portions of an antibody's structure responsible for its antigen binding affinity, to bind any specified antigen epitope.

However, while these protein design methods share common features and use the core IPRO calculation engine, the developed codes were typically incompatible with one another and cumbersome to use. The IPRO Suite shares a common, modular core of code that allows for full-integration of current (and future) protein engineering methods under a single user-friendly interface. The current implementation of the IPRO suite of programs contains IPRO, OptGraft, OptCDR, OptZyme, the modular antibody parts (MAPs) database[42] for antibody

[a] *R. J. Pantazes*
Chemical Engineering Department, University of California, Santa Barbara, 3357 Engineering II, Santa Barbara, California, 93106

[b] *M. J. Grisewood, T. Li, N. P. Gifford, C. D. Maranas*
Chemical Engineering Department, Pennsylvania State University, 112 Fenske Laboratory, University Park, Pennsylvania, 16802
E-mail: costas@psu.edu

Authors contributions: R.P. and M.G. developed most of the code and wrote the manuscript. T.L. contributed to developing the code. N.G. reviewed the use of the code. C.M. conceived of the project, provided oversight, reviewed and edited the manuscript. All authors read and approved the final manuscript.
Robert J. Pantazes and Matthew J. Grisewood are co-first authors

Contract grant sponsor: National Science Foundation; Contract grant number: CBET-0967062, CBET-1133040

structure prediction, and a Mutator program for predicting the properties of specific protein mutants.

The programs contained within the IPRO Suite incorporate largely the same definitions and concepts, allowing for the integration of the methods. Within the IPRO suite of programs, proteins are permitted to mutate so as to optimize binding to a particular ligand. The protein molecules that are permitted to mutate are referred to as design molecules (DMs) and the molecules (i.e., proteins, peptides, or small ligands) that they bind are referred to as target molecules (TMs). Within each DM, the exact residues that are allowed to mutate are referred to as design positions (DPs). Multiple DMs and TMs may be involved in a complex binding event. The totality of the participating molecules interacting with a particular binding topology is referred to as a binding assembly (BA). In lieu of binding energy ($= G_{BA,min} - G_{DMs,min} - G_{TMs,min}$) in IPRO, interaction energy ($= G_{BA,min} - G_{DMs} - G_{TMs}$) is used instead as it is less dependent on the force field used and requires fewer calculations. These concepts are used throughout the IPRO suite of programs to define the computational task underpinning the protein design goal. The Methods section provides a detailed description of the algorithmic infrastructure, input requirements for IPRO, and results on a test system for two of the programs. Finally, the Conclusions section summarizes the ability to consolidate various IPRO programs into a single interface.

Methods

Installation

The IPRO suite of programs may be installed by downloading the compressed folder from http://maranas.che.psu.edu. Once the compressed folder is downloaded, it must be moved to the directory where the software will be installed. As the IPRO suite of programs runs using the UNIX operating system, a file transfer protocol (e.g., Cyberduck) may be required to move the compressed folder to the proper directory. Next, the folder can be decompressed by entering "unzip IPRO_Suite.zip" on the command line. Prior to executing a case study, move into the "IPRO_Suite" directory and run "install.py" (i.e., "python install.py" entered to the command line) to finish the installation of the software. Force field inputs must be copied to the "input_files" folder and initial structures must be copied to the "structures" folder. Finally, a case study can be initiated by executing "Start_Experiment.py". This program will then identify additional experiment inputs from the user and begin the calculations.

IPRO architecture

The IPRO suite of programs is a collection of programs, files, and folders developed in UNIX. The programs are primarily written in Python and C++. The IPRO suite of programs currently only supports the use of the CHARMM[43] force field and the PDB[44] file format. Additional force fields and molecular structure formats will be supported in the future.

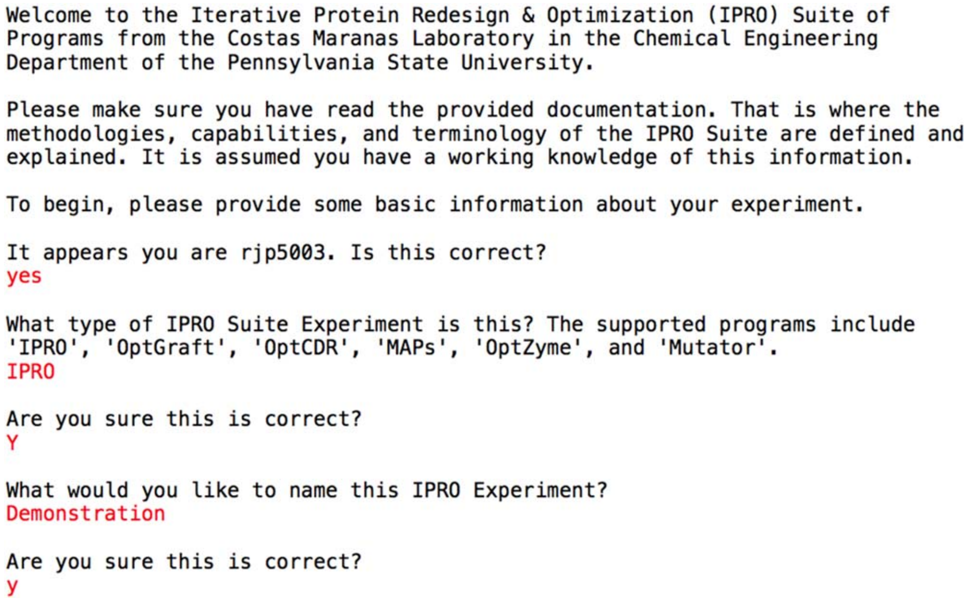
The contents of the IPRO suite folder include seven folders and the Python user interface program, "Start_Experiment.py". The "databases" folder is a repository for the MAPs database and the canonical structures used in OptCDR. "Experiments" is a folder where individual IPRO computational runs are stored. The "input_files" folder is where various input files (e.g., topology and parameter files for CHARMM) are stored. "Modules" stores the Python and C++ functions that are needed for running the programs of the IPRO Suite, which are stored in the "programs" folder. "References" contains information related to the IPRO suite of programs (e.g., the positions of the CDRs in OptCDR) as well as user interface documentation, and "structures" is where files of molecules' coordinates are stored.

All user inputs to IPRO are handled by the "Start_Experiment.py" program. This program asks the user for all of the information needed to run a particular type of IPRO Suite experiment, validates that the information is acceptable as it is collected, and starts the calculations. Figure 1 illustrates how the "Start_Experiment.py" program begins: by identifying the user, the type of IPRO computational task, and a unique name for the case study. A folder with this unique name is created in the "experiments" folder of the IPRO Suite and all calculation results are stored there.

Once the basic information of the experiment has been provided, the user is next asked to identify the DMs and TMs for the computational run. DMs are the proteins the user is redesigning to have modified interactions with the TMs. To clarify this terminology, consider the hypothetical situation of mutating the heavy chain of the antibody in PDB: 3KR3[45] to improve binding to insulin-like growth factor-II (IGF-2) while preventing binding to insulin-like growth factor-I (IGF-1) [PDB: 1IMX],[46] as illustrated in Figure 2. In this case, the heavy and light chains of the antibody are DMs, while IGF-1 and IGF-2 are denoted as TMs.

After all relevant molecules are specified, information regarding the force field is requested (i.e., input files and energy terms). The amino acid mutation step of IPRO requires the use of pairwise additive energy functions; however, the generalized Born with molecular volume integration method used for implicit solvation in CHARMM[47] is not pairwise additive. Therefore, if the user elects to use implicit solvation, they must also provide input files for the pairwise additive Lazaridis–Karplus implicit solvation method.[48]

Next, the "Start_Experiment.py" program checks the provided DM and TM structures for completeness (i.e., no missing residues) and compatibility with the force field and implicit solvation input files. Once this step is completed, the user is prompted to provide information specific to the computational task. This includes the selection of DPs and BAs. A DP is a residue in a DM that is permitted to mutate to modify the interaction energy with the TMs. Note that a DM may have no DPs, as is the case for the light chain in Figure 2. Interaction energy is defined as the energy of the minimized complex minus the energy of each molecule in the BA, assuming that each molecule's conformation is identical between the bound and free states. A BA is an interaction scenario for a set of TMs with the DMs investigated in the current computational study. For example, in Figure 2, one BA is the TM IGF-2 with the



```
Welcome to the Iterative Protein Redesign & Optimization (IPRO) Suite of
Programs from the Costas Maranas Laboratory in the Chemical Engineering
Department of the Pennsylvania State University.

Please make sure you have read the provided documentation. That is where the
methodologies, capabilities, and terminology of the IPRO Suite are defined and
explained. It is assumed you have a working knowledge of this information.

To begin, please provide some basic information about your experiment.

It appears you are rjp5003. Is this correct?
yes

What type of IPRO Suite Experiment is this? The supported programs include
'IPRO', 'OptGraft', 'OptCDR', 'MAPs', 'OptZyme', and 'Mutator'.
IPRO

Are you sure this is correct?
Y

What would you like to name this IPRO Experiment?
Demonstration

Are you sure this is correct?
y
```

Figure 1. Starting an IPRO suite of programs experiment. All IPRO suite of programs experiments can be created and run by the "Start_Experiment.py" program in the main IPRO Suite folder. In this figure, system messages and questions are shown in black, and corresponding example answers are shown in red. Messages, such as the one shown above reminding the user to read the documentation, are printed to the screen. Additionally, the user is prompted with questions that will ensure that the experiment is executed properly. Several of these are "yes" or "no" questions, and the program can handle various capitalization patterns and one-letter abbreviations.

heavy and light chain DMs, and the second BA is IGF-1 with the light and heavy chains. In the first BA, the objective is to improve binding between the DMs and TM, while the objective for the second BA is to suppress binding between the DMs and TM. IPRO consists of four ways in which it can modify the interactions between DMs and TMs in a BA: binding may be continuously improved, maintained above or below its initial value, or constantly suppressed (see Fig. 3). Note that while the amino acid sequence remains constant across all BAs, the corresponding rotamers may vary.

Unless otherwise stated, all settings listed are default values that may be adjusted by the user when beginning an experiment. The default values are stored in the "STANDARDS.py" file in the "modules" folder. Figure 4 shows the start of the file. On installation of the IPRO suite of programs, certain portions of this file must be edited (e.g., the InstallFolder parameter must be the global system path of the IPRO Suite folder), and the default settings may be changed to match the user's expected defaults. These specifications are stored in a "README.txt" file in the "references" folder.

Once the user has provided all of the necessary information, the case study folder is set up. Case study directories contain three folders ("structures," "input_files," "results"), two text files ("Experiment_Details.txt," "NAME_Summary.txt"), and one Python program (e.g., "IPRO.py"). The "structures" and "input_files" directories respectively contain the necessary molecule coordinates and input files. The "results" folder is where the different generated structures are stored. All generated structures use the residue numbering of their input structure files. The "Experiment_Details.txt" file stores all of the information the user provided about how to run the IPRO suite of programs experiment. It is intended to be both human and machine readable, and the start of this file is shown in Figure 5. On creation of the "Experiment_Details.txt" file, the user can choose whether or not to begin the experiment. The "NAME_Summary.txt" file, where NAME is the name of the experiment (i.e., Demonstration), keeps a record of the calculations performed, including when they begin, how long they take, and what was done in each iteration of the

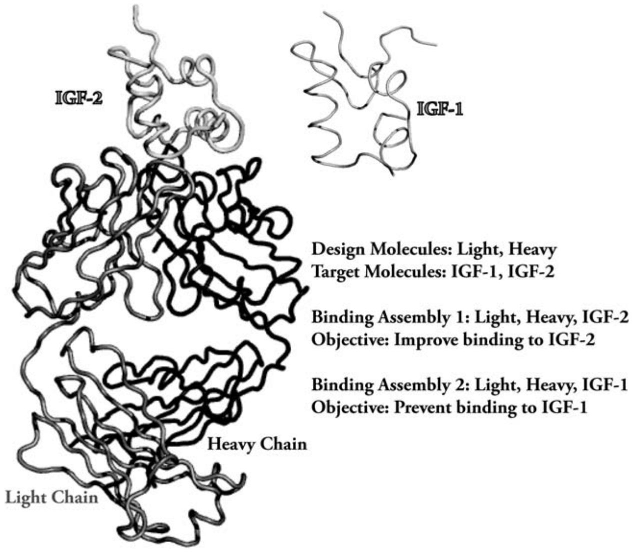

IGF-2
IGF-1
Design Molecules: Light, Heavy
Target Molecules: IGF-1, IGF-2
Binding Assembly 1: Light, Heavy, IGF-2
Objective: Improve binding to IGF-2
Binding Assembly 2: Light, Heavy, IGF-1
Objective: Prevent binding to IGF-1
Heavy Chain
Light Chain

Figure 2. Example demonstrating IPRO terminology. An antibody–IGF-2 complex [PDB: 3KR3] is hypothetically being designed to have improved binding to IGF-2, while preventing binding to IGF-1. The heavy and light chains of the antibody are DMs, and IGF-1 and IGF-2 are TMs in this example. Binding to IGF-1 and IGF-2 should be evaluated separately, so there are two BAs. Each BA contains all of the DMs and has a separate binding objective.

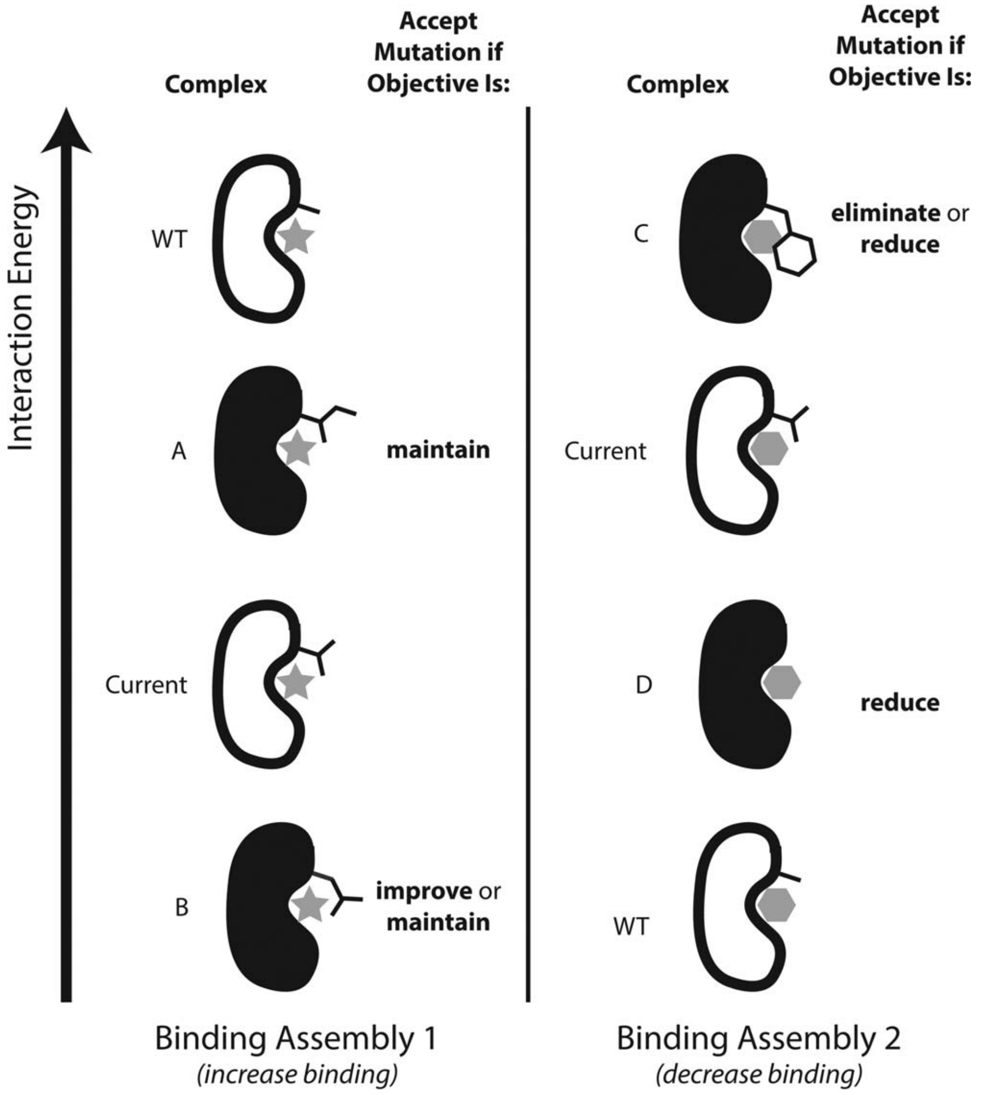


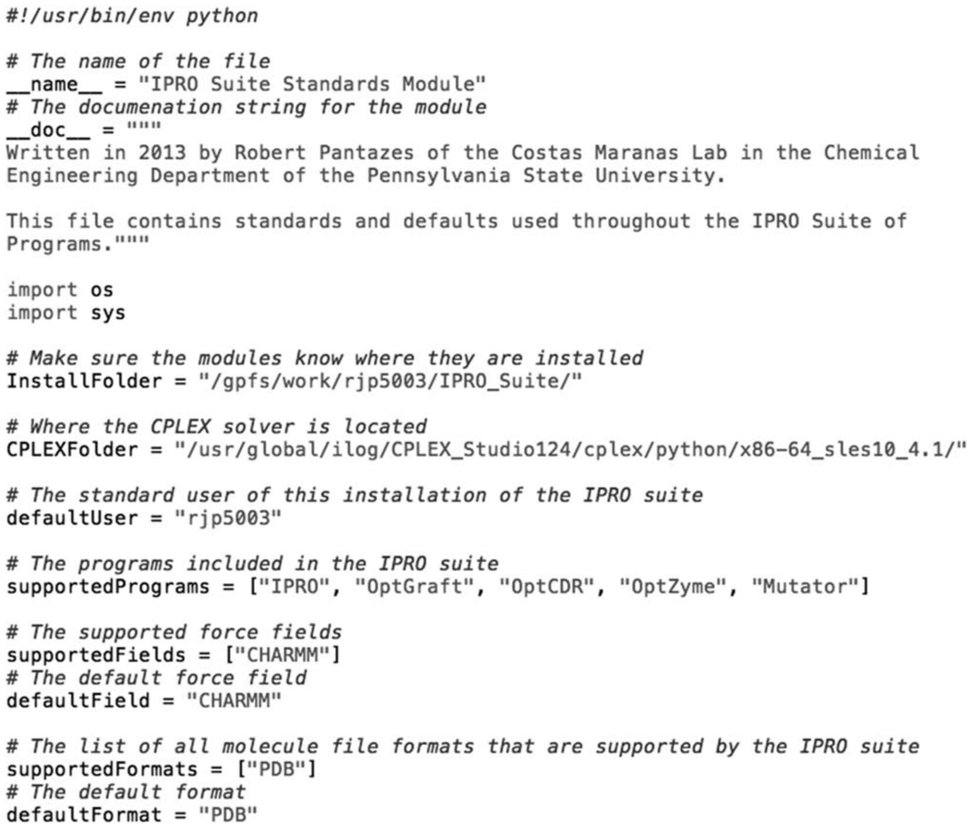
Figure 3. BA objective options. The figure illustrates the difference between the various BA objective choices. We consider two hypothetical BAs: improving binding to the first (TM: star) and worsening binding for the second (TM: hexagon). Four theoretical mutant DMs (A–D) are shown in black. Reference DMs, wildtype (WT) and the "best" mutant identified prior to the current iteration (Current), are depicted in white. If mutant "A" is generated in a given IPRO iteration, it can only be considered "best" if the first BA objective is "maintain". That is, if the first BA objective is "improve," "reduce," or "eliminate," then mutant "A" will be discarded. Similarly, mutant "B" may only be considered the "best" if the first BA objective is "improve" or "maintain," mutant "C" if the second BA objective is "eliminate" or "reduce," and mutant "D" if the second BA objective is "reduce."

calculation. Finally, the IPRO Suite program (i.e., "IPRO.py") carries out the calculations.

IPRO calculation engine

At the core of the IPRO suite of programs is the IPRO calculation engine. The original implementation of IPRO suggested stabilizing mutations to a combinatorial library of recombined proteins, with the limiting case being the redesign of a single protein.[37] Only the latter functionality is retained in this new release. The six basic steps of an IPRO iteration are shown in Figure 6. First, a randomly selected set of contiguous residues is selected in a DM, and the backbone phi and psi dihedral angles are randomly perturbed. Subsequently, residues in and around the perturbed region are repacked and mutated using a rotamer library[49] and mixed-integer linear programming (MILP) optimization. The third step is a local, rigid-body redocking of the TMs. This is followed by a force field complex energy minimization (using the Adopted Basis Newton-Raphson method)[50] and energy calculation. The calculated energy between the protein and substrate is used in the Metropolis criterion to determine whether or not to retain the new protein as the starting point for the next iteration. Using these six steps over the course of thousands of iterations, IPRO can identify combinations of backbone and amino acid changes that improve the desired performance metrics.

In the first step of IPRO, a random backbone perturbation to a DM is carried out. This begins by randomly selecting a DP. Next, the total size of the perturbed region (one to five residues) and the relative position of the DP within this contiguous stretch of residues are randomly chosen. For each of the perturbed residues, the phi and psi dihedral angles are



```
#!/usr/bin/env python

# The name of the file
__name__ = "IPRO Suite Standards Module"
# The documenation string for the module
__doc__ = """
Written in 2013 by Robert Pantazes of the Costas Maranas Lab in the Chemical
Engineering Department of the Pennsylvania State University.

This file contains standards and defaults used throughout the IPRO Suite of
Programs."""

import os
import sys

# Make sure the modules know where they are installed
InstallFolder = "/gpfs/work/rjp5003/IPRO_Suite/"

# Where the CPLEX solver is located
CPLEXFolder = "/usr/global/ilog/CPLEX_Studio124/cplex/python/x86-64_sles10_4.1/"

# The standard user of this installation of the IPRO suite
defaultUser = "rjp5003"

# The programs included in the IPRO suite
supportedPrograms = ["IPRO", "OptGraft", "OptCDR", "OptZyme", "Mutator"]

# The supported force fields
supportedFields = ["CHARMM"]
# The default force field
defaultField = "CHARMM"

# The list of all molecule file formats that are supported by the IPRO suite
supportedFormats = ["PDB"]
# The default format
defaultFormat = "PDB"
```

Figure 4. The "STANDARDS.py" file. Located in the IPRO Suite "modules" folder, "STANDARDS.py" is the file storing the default settings for the IPRO suite of programs. The beginning of that file is shown here. The "Start_Experiment.py" program accesses this information to provide the user with suggested values to its extensive questions. When the IPRO Suite is installed, users should modify the contents of this file to match their expected default settings. For example, the defaultUser, "rjp5003," should be changed to the appropriate user's name.

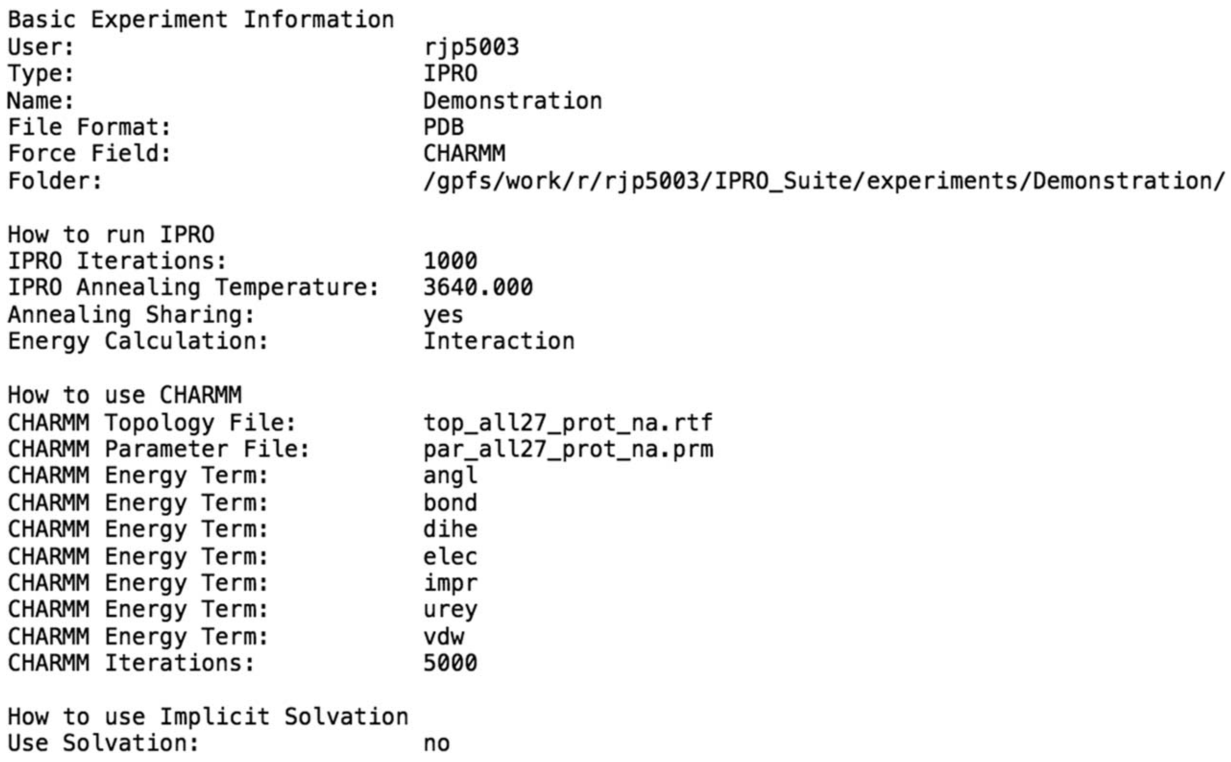
randomly perturbed using a Gaussian distribution centered at zero degrees with a standard deviation of 1.5 degrees. No modification greater in magnitude than five degrees is permitted. Five residues on each side of the perturbed region are free to move during the perturbation to prevent the dihedral angle changes from causing long-range structural clashes. These 11–15 residues are mutated to glycine and an energy minimization with strong (i.e., 32,800 kcal mol^{-1} radian^{-2}) restraints on the perturbed dihedral angles is carried out to create the perturbed DM structure. Residues outside the 11–15 residues are fixed in place during the perturbation. Each BA undergoes the same random perturbation.

The second step of IPRO is the use of a rotamer library and MILP optimization to repack the amino acid side chains in and around the perturbed region. The user has the option of also repacking residues that are spatially nearby the perturbed region. The default definition of closeness is having at least one heavy atom (i.e., not hydrogen) within 4.5 Å of a heavy atom in a perturbed residue.[10] Only DPs within the perturbed region are permitted to mutate. By default, all DPs can mutate to any amino acid, but the user may specify the allowable amino acid mutations for each DP.

The rotamer library[49] does not contain proline rotamers. Therefore a proline rotamer was created to allow for the option of mutating to proline. One thousand proline residues were collected from a database of antibodies,[42] and the proline with the smallest average heavy-atom root mean-squared deviation (RMSD) with the other residues was selected to serve as the rotamer. This was residue 14 in molecule chain B of PDB: 1CFQ.[51] This rotamer is only used when it is sterically permitted, as determined by an analysis of literature data,[52] or when a residue is already a proline and is not mutating.

The rotamer–nonrotamer (RNR; i.e., the TMs and all parts of the DMs that are not being replaced by rotamers) and rotamer–rotamer (RR) energies are calculated using the pairwise additive, nonbonded energy terms from the force field (i.e., Van der Waals [VDW], electrostatics)[43] and implicit solvation. The IPRO Suite allows the VDW repulsion term to be "softened" when calculating the RR and RNR energies.[53] The rotamer library often contains terms that attempt to match observed statistics.[49] The user may choose to include these terms in the RNR energies, but the default is to be excluded.

When selecting the rotamers, the RNR pairwise interaction energies are calculated first. Next, the total number of rotamers is reduced below a threshold of 1400 to ensure that the MILP problem can be solved in a reasonable amount of time (i.e., $<$10 min) and memory usage (i.e., $<$ 4 GB). This is performed by retaining the three lowest RNR energy rotamers of each permitted kind of amino acid at each position. Additionally, all rotamers that are within 40 kcal mol^{-1} of the lowest RNR energy rotamer at each position are retained. The parameters used to discard rotamers (i.e., three of each amino acid and a 40 kcal mol^{-1} energy window) progressively



```
Basic Experiment Information
User:                          rjp5003
Type:                          IPRO
Name:                          Demonstration
File Format:                   PDB
Force Field:                   CHARMM
Folder:                        /gpfs/work/r/rjp5003/IPRO_Suite/experiments/Demonstration/

How to run IPRO
IPRO Iterations:               1000
IPRO Annealing Temperature:    3640.000
Annealing Sharing:             yes
Energy Calculation:            Interaction

How to use CHARMM
CHARMM Topology File:          top_all27_prot_na.rtf
CHARMM Parameter File:         par_all27_prot_na.prm
CHARMM Energy Term:            angl
CHARMM Energy Term:            bond
CHARMM Energy Term:            dihe
CHARMM Energy Term:            elec
CHARMM Energy Term:            impr
CHARMM Energy Term:            urey
CHARMM Energy Term:            vdw
CHARMM Iterations:             5000

How to use Implicit Solvation
Use Solvation:                 no
```

Figure 5. The contents of an "Experiment_Details.txt" file. The "Experiment_Details.txt" file contains all of the information the user provided for how to run a particular IPRO suite of programs experiment. It is intended to be both a human-readable record of how the experiment was conducted as well as instructions for the IPRO suite of programs on how to carry out the experiment.

become more stringent until the number of rotamers kept is below the threshold, and then the RR pairwise interaction energies are calculated.

Once all RNR and RR energies are calculated, a MILP formulation is used to select the minimum energy arrangement of rotamers. This requires the definition of the following sets: the positions of residues receiving rotamers: $i, j=1,\ldots,N$, and the rotamers at position i: $r, s=1,\ldots R_i$. The interaction energy of rotamer r at position i with the nonrotamer portion of the BA is stored in parameter $EC_{i,r}$ and the interaction energy between rotamer r at position i and rotamer s at position j is stored in parameter $ER_{i,r}^{j,s}$. Binary variable $y_{i,r}$ is equal to one if rotamer r is selected at position i and zero otherwise. Similarly, binary variable $w_{i,r}^{j,s}$ is equal to one if rotamer r at position i and rotamer s at position j are selected simultaneously and zero otherwise. Using these sets, parameters, and variables, the optimal combination of rotamers can be selected by minimizing:

$$\sum_{i=1}^{N}\sum_{r=1}^{R_{i,}} y_{i,r}EC_{i,r}+\sum_{i=1}^{N-1}\sum_{r=1}^{R_i}\sum_{j=i+1}^{N}\sum_{s=1}^{R_j} w_{i,r}^{j,s}ER_{i,r}^{j,s} \qquad (1)$$

Equation (1) is the objective function of the MILP and minimizes the net energy of the selected rotamers with each other and with the nonrotamer portion of the BA. This equation is subject to:

$$\sum_{r=1}^{R_i} y_{i,r}=1\ \ \forall i|1,\ldots,N \qquad (2)$$

Equation (2) ensures that exactly one rotamer is selected at each position.

$$w_{i,r}^{j,s}=y_{i,r}y_{j,s}\ \ \forall i|1,\ldots,N-1,\forall r|1,\ldots,R_i,\ \ j|i+1,\ldots,N,\forall s|1,\ldots,R_j \qquad (3)$$

Equation (3) ensures that $w_{i,r}^{j,s}$ only assumes a value of one if both $y_{i,r}$ and $y_{j,s}$ have values of one. However, eq. (3) generate $\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}R_iR_j$ bilinear constraints. It can be equivalently be

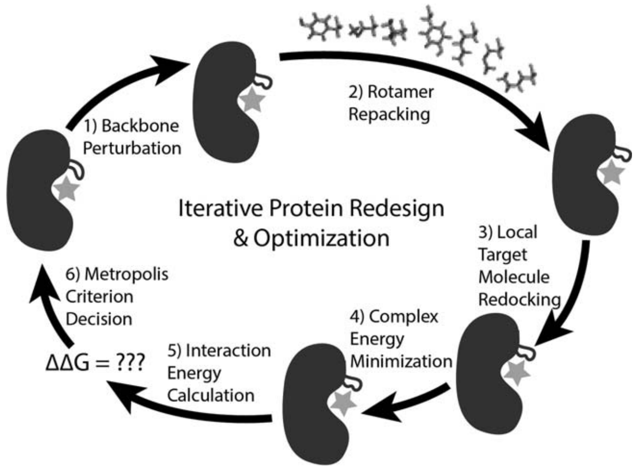


Figure 6. The steps of an IPRO iteration. In this visualization of a single IPRO iteration, the DM is shown in dark grey and the TM is shown as a light grey star. In the first step of IPRO, a randomly selected portion of the backbone is perturbed as indicated by the altered loop conformation. The optimal set of rotamers for this new conformation is then selected. Next, the TM is allowed be reoriented within the active site as indicated by the rotated TM. The complex energy of the system is then minimized and the results of the IPRO iteration are analyzed to decide if they should be retained.

recast in a linear form while reducing the total number of constraints by a factor of $\sum_{i=1}^{N-1}\sum_{j=i+1}^{N}(R_iR_j)/\left((N-1)\sum_{i=1}^{N}R_i\right)$ as shown in eqs. (4) and (5).[37] For example, if there were 21 rotamers at position $i = 1$, 29 rotamers at position $i = 2$, and 15 rotamers at position $i = 3$ in a tripeptide, there would be 1359 (= 21 × 29 + 21 × 15 + 29 × 15) bilinear constraints and 130 [= 2 × (21 + 21 + 15)] linear constraints, approximately an order of magnitude difference.

$$y_{i,r}=\sum_{s=1}^{R_j} w_{i,r}^{j,s} \quad \forall i|1,\ldots,N-1, \forall j|i+1,\ldots,N, \ \forall r|1,\ldots,R_i \qquad (4)$$

$$y_{j,s}=\sum_{r=1}^{R_i} w_{i,r}^{j,s} \ \forall i|1,\ldots,N-1, \ \forall j|i+1,\ldots,N, \ \forall s|1,\ldots,R_j \qquad (5)$$

Equations (1), (2), (4), and (5) form the optimal rotamer selection MILP and can be solved to optimality using the CPLEX solver.[54] The IPRO suite of programs supports calling the CPLEX solver directly from Python or through the general algebraic modeling system (GAMS)[55] interface. The optimal combinations of rotamers are selected sequentially for each BA whereas residue identity remains the same for all BAs.

The third step of an IPRO iteration is a local, rigid-body redocking of the TMs. Docking can be time-consuming so this step is only performed once for every three iterations. Docking uses the same energy functions as the rotamer selection step. During docking, a TM is randomly perturbed along and around the *X*, *Y*, and *Z* axes using a Gaussian distribution centered at zero, with standard deviations of 0.2 Å and 2.0°, respectively. After the TM is perturbed, the net interaction energy of all of the TMs with each other and with the DMs is evaluated, and the movement of the TM is retained if the net interaction energy has improved. Each TM is randomly perturbed and 500 iterations are performed. A constant cooling schedule is used for simulated annealing. The temperature at the start (3640 K) and end (2190 K) of docking retain 25 and 10%, respectively, of positions within 10 kcal mol^{-1} of the best solutions thus far.

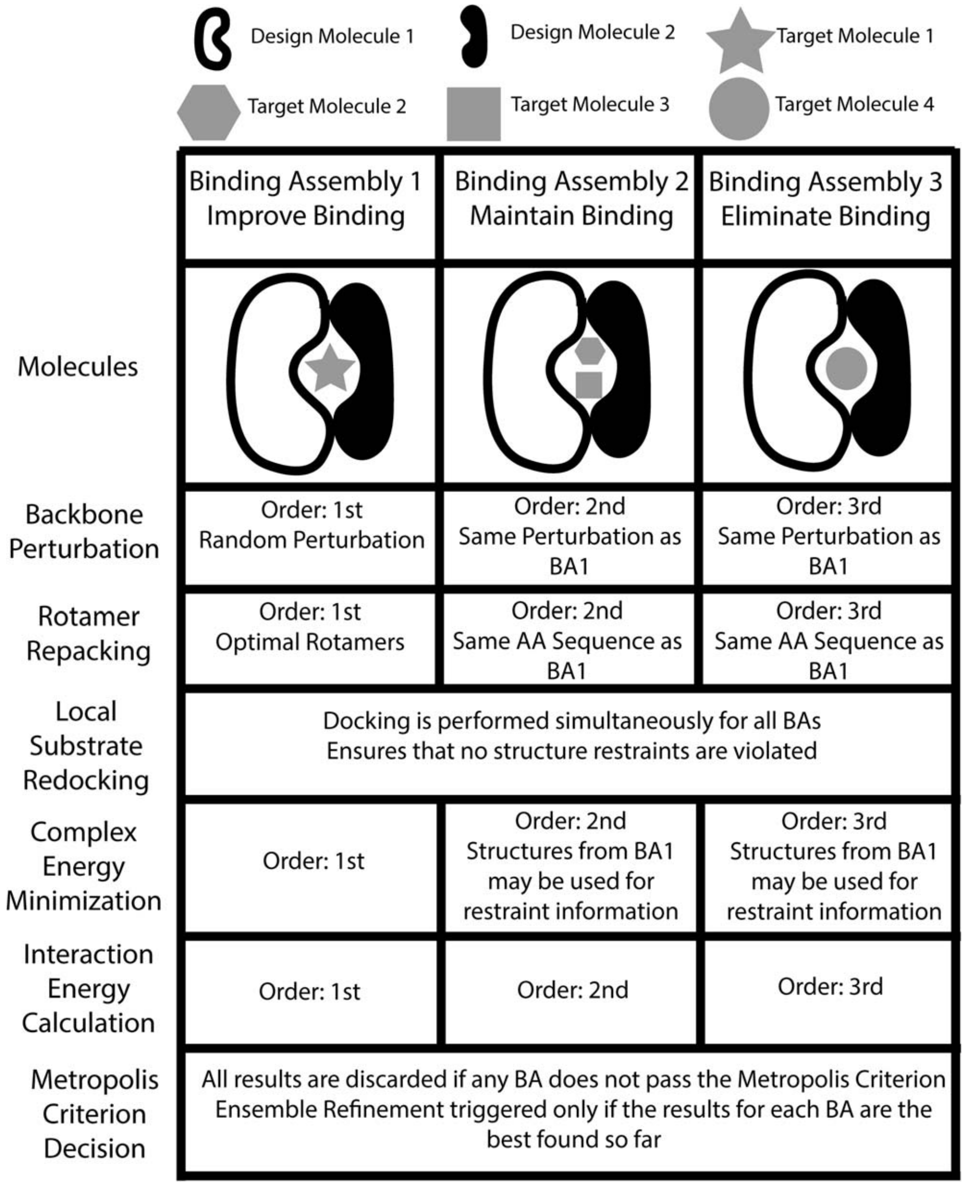
The fourth and fifth steps of an IPRO iteration are a force field complex energy minimization and interaction energy calculation, respectively. Finally, the calculated interaction energy is used in the Metropolis criterion to determine whether or not to retain the results of the IPRO iteration. The temperature used in the Boltzmann factor is 3640 K, which retains 25% of designs within 10 kcal mol^{-1} of the best design. All BAs must pass the selection criteria for the results of the iteration to be retained. The six basic steps of an IPRO iteration (Fig. 6) are illustrated using example BAs in Figure 7. The following subsections list IPRO features that were unavailable (i.e., ensemble structure refinements, parallelization, and OptCDR canonical structures database) or less efficient (i.e., OptGraft binding pocket selection MILP) in previous versions.

Algorithm improvements

Ensemble Structure Refinements. The result of an IPRO design iteration is a specific backbone conformation and amino acid assignment that minimizes the objective function of the MILP described in Step 2. This energy minimum may not be the true global energy minimum as it is dependent on the backbone geometry assumed after the sequence of random perturbation steps. To remedy this dependence of the identified interaction energy-minimized structures on the history of traversed random backbone perturbations, IPRO allows the use of optional ensemble structure refinements. These refinements are performed by generating an ensemble of 10 structures, each of which is generated from 25 "refinement" iterations of IPRO. During refinement iterations, the objective for every BA is to minimize its complex energy, and the energy optimization goals of the BAs are not considered. In addition, no mutations are permitted during refinement iterations. In essence, refinement iterations are searching for improved local energy minima for a specific DM amino acid sequence. After all refinement iterations have been completed, the average BA objective values of the refinement ensemble are evaluated to determine whether or not a particular design is actually the best identified so far. This ensemble approach in evaluating energies has shown a high correlation (R^2 = 0.960) between calculated interaction energies and experimentally measured binding affinities in previous work.[40] If structure refinements are being performed, there is the option of using binding energy instead of interaction energy to evaluate BA objective values. Binding energies are calculated after generating an additional BA that exclusively contains DMs. Analysis during the development of the IPRO suite of programs revealed that allowing binding energy calculations without structure refinements resulted in highly inconsistent results between different repetitions of the same experiment.

Parallelization. In contrast to earlier versions, the IPRO suite of programs allows for separate processors to simultaneously work on the same IPRO Suite experiment. An independent processor carries out each IPRO iteration. The results from each iteration are communicated between processors. The user has the additional option to share results between processors that were retained using the Metropolis criterion. If one processor completes an iteration and determines that an ensemble structure refinement is necessary, a system flag is created that will cause other processors to delete their ongoing iterations and instead participate in the ensemble structure refinement. In the following, a brief description of the optimization details related with transferring a binding pocket onto a new protein scaffold (i.e., OptGraft) and the design of complementary determining regions for improving affinity for a given antigen are described.

OptGraft Binding Pocket Selection MILP. OptGraft inserts a binding site from one protein into a novel host and then uses IPRO to suggest mutations that improve the geometry around the binding pocket. The selection of the host residues for the binding pocket in the novel protein is performed using a MILP optimization formulation that has been modified from its original implementation[39] to solve faster. The MILP requires the sets defining the residues in the novel host protein: $i,j=1,\ldots,N$, and the residues of the binding pocket in the original "donor" protein: $k,l=1,\ldots,K$. The MILP makes decisions based on Cα–Cα and Cβ–Cβ distances. The Cα–Cα and Cβ–Cβ distances in the novel "host" protein are stored in parameters $RA_{i,j}$ and $RB_{i,j}$, respectively. These distances between the atoms in the binding site residues are stored in parameters $ra_{k,l}$ and $rb_{k,l}$. Binary



	Binding Assembly 1 Improve Binding	Binding Assembly 2 Maintain Binding	Binding Assembly 3 Eliminate Binding
Molecules			
Backbone Perturbation	Order: 1st Random Perturbation	Order: 2nd Same Perturbation as BA1	Order: 3rd Same Perturbation as BA1
Rotamer Repacking	Order: 1st Optimal Rotamers	Order: 2nd Same AA Sequence as BA1	Order: 3rd Same AA Sequence as BA1
Local Substrate Redocking	Docking is performed simultaneously for all BAs Ensures that no structure restraints are violated		
Complex Energy Minimization	Order: 1st	Order: 2nd Structures from BA1 may be used for restraint information	Order: 3rd Structures from BA1 may be used for restraint information
Interaction Energy Calculation	Order: 1st	Order: 2nd	Order: 3rd
Metropolis Criterion Decision	All results are discarded if any BA does not pass the Metropolis Criterion Ensemble Refinement triggered only if the results for each BA are the best found so far		

Figure 7. Example of multiple binding assemblies during an IPRO iteration. IPRO uses BAs to simultaneously evaluate multiple objectives. Consider the hypothetical case of an enzyme formed from two separate polypeptides with the objective to change specificity from one substrate to another and the additional constraint that the enzyme must simultaneously bind two cofactors to recharge after each catalytic event. This problem has three distinct binding objectives, shown in the three BAs for this figure. Note that the DMs are shown in black and white, the favored TM as a star, the disfavored TM as a circle, and the two cofactor TMs as a hexagon and a square. Also note that each BA must contain all DMs. The first BA's objective is to "improve" binding to the favored TM. The second BA's objective is to "maintain" binding to the necessary cofactors (i.e., not allowing binding to worsen, but also not requiring it to improve). Finally, the objective of the third BA it to "eliminate" binding to the disfavored TM. The steps of IPRO are listed from top to bottom, and the figure shows the relative ordering of the BAs within each step of IPRO.

variable $Y_{i,k}$ is equal to one if residue i in the novel host protein is selected as the equivalent to residue k in the binding site and zero otherwise. Similarly, binary variable $W_{i,k}^{j,l}$ is equal to one if residue i is selected as the equivalent to residue k and simultaneously residue j is selected as the equivalent of residue l; otherwise, $W_{i,k}^{j,l}$ is zero. The MILP can then be written to minimize:

$$\sum_{k=1}^{K-1}\sum_{i=1}^{N}\sum_{l=k+1}^{K}\sum_{j=1}^{N}\left[\left(RA_{i,j}-ra_{k,l}\right)^2+\left(RB_{i,j}-rb_{k,l}\right)^2\right]W_{i,k}^{j,l} \qquad (6)$$

Equation (6) minimizes the sum of the squared differences in distances between the α and β carbons in the selected binding site from the original binding site, thus maintaining the original geometry. This equation is subject to:

$$\sum_{i=1}^{N} Y_{i,k}=1 \qquad \forall k|1,\ldots,K \qquad (7)$$

This equation ensures that every residue in the binding site is assigned to exactly one residue in the "host" protein.

$$\sum_{k=1}^{K} Y_{i,k}\leq 1 \qquad \forall i|1,\ldots,N \qquad (8)$$

Equation (8) makes certain that no residue is assigned as more than one residue from the binding site.

$$W_{i,k}^{j,l} = Y_{i,k}Y_{j,l} \quad \forall i|1,\dots,N, \forall j|1,\dots,N, \forall k|1,\dots,K-1, \forall l|k+1,\dots,K \tag{9}$$

This establishes the relationship between $W_{i,k}^{j,l}$, $Y_{i,k}$, and $Y_{j,l}$, which is nonlinear and must be linearized. The modification from the original formulation[39] is through linearization of eq. (9) in a more efficient manner. This starts by summing over the i positions in the novel host protein on both the left and right hand sides of eq. (9).

$$\sum_{i=1}^{N} W_{i,k}^{j,l} = \sum_{i=1}^{N} Y_{i,k}Y_{j,l} \;\; \forall j|1,\dots,N, \;\; \forall k|1,\dots,K-1, \;\; \forall l|k+1,\dots,K \tag{10}$$

$Y_{j,l}$ is independent of the index i being summed over in eq. (10) and can be moved outside of the summation on the right hand side. Combining this with eq. (7) gives:

$$Y_{j,l} = \sum_{i=1}^{N} W_{i,k}^{j,l} \;\; \forall j|1,\dots,N, \;\; \forall k|1,\dots,K-1, \;\; \forall l|k+1,\dots,K \tag{11}$$

Equation (12) results from the same procedure as eq. (11), when a summation over j is used instead.

$$Y_{i,k} = \sum_{j=1}^{N} W_{i,k}^{j,l} \;\; \forall i|1,\dots,N, \;\; \forall k|1,\dots,K-1, \;\; \forall l|k+1,\dots,K \tag{12}$$

Together, eqs. (11) and (12) linearize eq. (9). Equations (6–8), (11), and (12) form the binding site selection MILP and can be solved to optimality by calling the CPLEX[54] solver directly from Python or using GAMS.[55]

OptCDR Canonical Structure Database. The set of antibody structures examined to make the MAPs[42] database had their CDRs collected and clustered to generate a new database of canonical structures that follow IMGT® unique numbering[56–59], from IMGT®, the international ImMunoGeneTics information system® (http://www.imgt.org).[60] The clustering was carried out using identical methods to the original OptCDR publication.[41] It was conducted such that at the end of clustering, all structures within a cluster had a backbone atom (N, Cα, C) RMSD of no more than 1.5 Å from the model structure of the cluster, where the model is the structure with the smallest average backbone atom RMSD with all other members of the cluster. All clusters for the H3 CDR were retained, and clusters with at least three members were retained for CDRs H1, H2, L1, L2, and L3. Several *a posteriori* modifications were made to reduce clashing among the canonical structure models. The old and new canonical structure databases are listed in Table 1. Of note is the significant increase in the number of L3 canonical structures. This is consistent with the expectation of structural diversity in this CDR due to the junction of a Variable and Joining gene introduced during V-(D)-J recombination.

Table 1. The OptCDR Canonical Structures.

	Original		New	
	Amino acids	Canonical structures	Amino acids	Canonical structures
H1	10–12	8	8–10	9
H2	9–12	4	7–10	5
H3	3–22	124	5–24	135
L1	10–17	13	5–12	13
L2	7	1	3	1
L3	8-11	9	7–13	16

Here are the number of amino acids and canonical structures for each CDR from the original OptCDR publication[41] and updated values using IMGT® unique numbering. The definition of each CDR is different in the two cases and approximately 300 additional antibodies were analyzed in the updated dataset. However, it is interesting to note that the number of L3 CDRs significantly increased. This is consistent with the expected structural diversity introduced by the gene junction between the variable and joining genes in this CDR during V-(D)-J recombination.

IPRO suite programs

IPRO. IPRO redesigns proteins to have modified binding interactions with one or more substrates. To run an IPRO experiment, the user must provide all of the information needed to run the experiment: the user's name, the name of the experiment, the DMs and TMs, how to use the force field, how to use implicit solvation, the DPs, the kinds of amino acids mutations permitted for each DP, the BAs, how to run docking, how to repack rotamers, the temperature for the Metropolis criterion, whether or not to share results retained by the Metropolis criterion between processors, the number of design iterations to run, and how to do ensemble structure refinements. In addition, the IPRO suite of programs permits four types of restraints on atom positions that are rigorously enforced throughout all functions: (1) atoms may be permanently fixed in place, (2) atoms may be kept near their positions in their initial structures, (3) two atoms in the same BA may be restrained to a specified distance, and (4) four atoms in the same BA may be restrained to a provided dihedral angle value.

Once the information has been provided and "Start_Experiment.py" has created the folder to run the experiment in, the calculations begin. Calculations start with a force field energy minimization of the initial BAs, replacement of all DM amino acids with the most similar rotamer from the rotamer library, and a second force field energy minimization of the BAs. This provides an initial starting point for the experiment. If ensemble structure refinements are being used, twice the number of refinement (i.e., 50) iterations are performed to further refine this initial structure. Once the initial structure is constructed, 3000 design iterations are run, with interruptions by ensemble structure refinements whenever a best result is found. Over the course of thousands of iterations, IPRO can identify combinations of structure perturbations and sequence mutations that provide improvements in the interactions between the DMs and TMs.

OptGraft. OptGraft inserts a binding site from one protein into a novel host and then uses IPRO to suggest mutations that improve the geometry around the pocket. An OptGraft experiment may be run after the user provides the same information as for an IPRO experiment, with the added necessity of identifying the binding site in one molecule and the particular DM serving as the scaffold. In addition, residues in the host

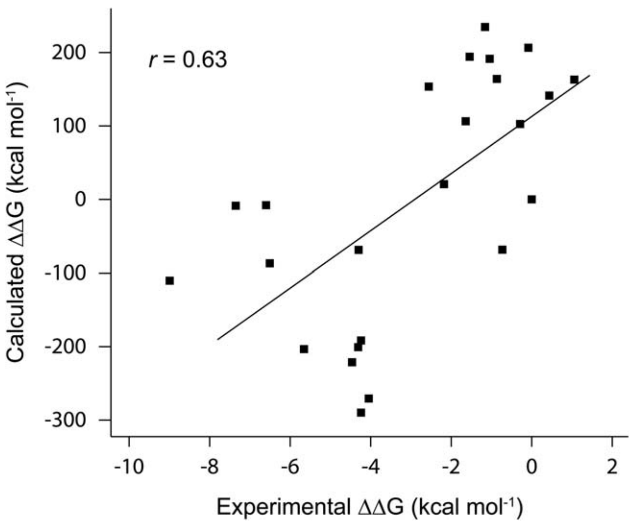


Figure 8. Comparison of Barnase-Barstar experimental data and Mutator results. The Mutator program was used to find the calculated $\Delta\Delta G$ ($= \Delta G_{\text{mutant}} - \Delta G_{\text{wildtype}}$), which is approximated within the program using interaction energies. Experimental $\Delta\Delta G$ values were obtained from SKEMPI. The Pearson product-moment correlation coefficient between the experimental and computational data was 0.63. The different scale of the calculated $\Delta\Delta G$ (approximately one of magnitude difference) suggests that these values are only qualitatively meaningful and cannot be used to approximate the true binding free energy. The slope for the best fit line is 38.85, and the *y*-intercept is 113.37 kcal mol^{-1}.

DM containing a heavy atom within 4.5 Å of a heavy atom in the new binding site are automatically selected as DPs. The user may exclude some or all of these DPs and select others, if desired. Also, strong distance restraints on the α and β carbon distances of the binding site residues are used throughout the OptGraft experiment to ensure the resulting design has the appropriate geometry. An OptGraft experiment begins by inserting the novel binding site into the designated DM and then proceeds in the same manner as an IPRO experiment.

OptCDR. OptCDR[41] is a method for the *de novo* design of antibody CDRs to bind any specified antigen epitope. It works through a four-step procedure: selecting a combination of canonical structures that is most likely to be able to bind an antigen's epitope based on geometric criteria, filling in the selected structures' amino acid sequence, using IPRO to computationally affinity-mature the complex, and finally generating a library of the most favorable mutants. In this way, OptCDR can generate a library of any specified size of antibodies that bind a target epitope.

To run an OptCDR experiment, the user must provide most of the same information as for an IPRO experiment. However, there is no specification of DMs or DPs as the antibody is designed from scratch, and every residue in the CDRs is permitted to mutate. In addition, the user must specify whether to design a heavy chain, a light chain, or both. If desired, OptCDR also now has the ability to include framework residues (i.e., the portions of the antibody variable domain that are not CDRs) during calculations. These framework residues are left in their initial configuration throughout OptCDR.

An OptCDR experiment begins by generating 3000 random initial placements for the antigen and identifying the best combination of canonical structures for each. The best positions are retained for 100 independent library designs. For each combination of canonical structures, the amino acids of the canonical structures are initialized from glycine one CDR at a time (due to memory limitations for the MILP). As described in the original OptCDR implementation,[41] restraints are imposed on which amino acids are allowed at each position in a CDR based on experimental observations of the canonical structure. In addition, the maximum percent usage of each amino acid and all charged amino acids is limited, based on the maximum percent usage observed in each CDR, using integer cuts in the rotamer selection MILP. After each CDR is initialized, the docking function of IPRO is run and the amino acids of each CDR are reassigned to create an initial antibody. Once this structure is generated, IPRO is used to computationally affinity mature the designed CDRs. During an IPRO iteration in OptCDR, a CDR is randomly selected and the entire backbone is randomly perturbed, instead of only perturbing a portion of it. Also, the sequence restraints already mentioned for initializing the CDR sequences are enforced in IPRO. Finally, the rotamer selection MILP is iteratively solved, using integer cuts to eliminate previous solutions, to generate a library of mutants for the affinity matured antibody design.

MAPs. The MAPs database is integrated into the IPRO suite of programs to predict antibody structures from their sequences, as described in the original publication.[42] Predicting an antibody's structure is the most simple program within the IPRO Suite. The user must provide their name, the type of experiment (i.e., MAPs), the experiment directory name, and a heavy and/or light variable domain sequence. The IPRO Suite automatically assigns germline genes to the antibody sequences, collects the most similar MAPs pieces and mutates them to the germline gene sequence, mutates this antibody structure to the input sequences, and runs a force field energy minimization to generate the final predicted antibody structure.

OptZyme. OptZyme[40] uses ground state substrates and transition state analogues in IPRO to predict mutations that improve catalytic properties of enzymes. Running an OptZyme experiment is identical to running an IPRO experiment, except the user must specify which TMs are the ground state substrates and which are the transition state analogues. If multiple catalytic reactions are being simultaneously considered (e.g., suppressing catalysis or binding of a competitor), the user must specify which ground state substrates and which transition state analogues are paired. This information is used to automatically generate BAs with the appropriate objectives.

Mutator. Often, it is desirable to computationally analyze specific protein mutants for properties, rather than try to design for those properties. To facilitate this, a Mutator program was created that uses the ensemble structure refinement function of the IPRO suite of programs. Running a Mutator experiment is identical to running an IPRO experiment, except instead of identifying DPs the user must identify the amino acid changes for the mutants. Each mutant may contain multiple amino acid substitutions, and multiple mutants to the same system

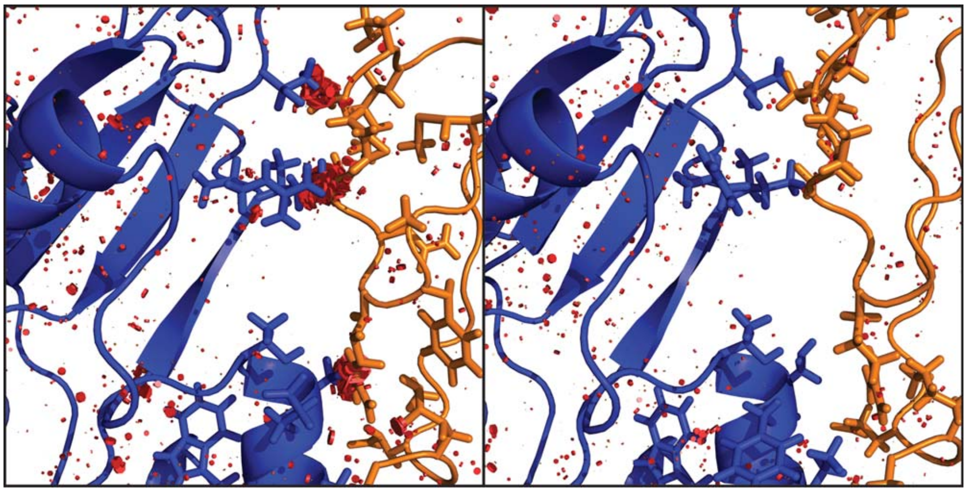


Figure 9. Improving barnase binding affinity for a non-native ligand using IPRO. IPRO was used to mutate Barnase (shown in blue) to better its interaction energy with pre-mRNA-splicing factor CWC26 (shown in orange). Barnase DPs and splicing factor residues within 4.5 Å of the DPs are shown as sticks. Steric clashes are shown as red disks. On the left (initial structure), there are three regions of significant steric hindrance, whereas on the right (final structure), these unfavorable contacts are absent. These contacts were removed through the use of IPRO to improve the interaction with the splicing factor. This figure was generated using PyMOL and the "show_bumps.py" script.

can be analyzed in a single experiment. The residue positions that are being mutated are treated as pseudo-DPs so that IPRO iterations can select regions of the structure to perturb. A Mutator experiment begins by executing a structure refinement for the original system of molecules. The refined complex is used as the starting point for each mutant. The optimal rotamer selection MILP is used to generate an initial version of the mutant, and an ensemble structure refinement is subsequently used to evaluate its properties.

Test system

The Barnase-Barstar system is used to illustrate the capabilities of the IPRO suite of programs. This system was chosen as Barnase and its inhibitor, Barstar, do not require further force field parameterization as both molecules are proteins, which are already parameterized for the CHARMM force field.[43] Furthermore, the structure of the protein-inhibitor complex was resolved at 2.0 Å (PDB entry 1BRS),[61] and the structural database of kinetics and energetics of mutant protein interactions (SKEMPI)[62] provides binding affinities for 23 Barnase mutants. We used the Mutator program to compare calculated interaction energies to known binding free energies from SKEMPI. Mutator was run using the default values for each parameter to evaluate the performance of the program without further fine-tuning. The calculated values correlated reasonably well (Fig. 8) with experimental binding affinities ($r = 0.63$). This correlation value is similar to the ones found using other computational methods ($r < 0.60$).[63] Results for this and other systems not described here demonstrate that Mutator is a useful tool for qualitatively assessing mutants with a reasonable level of accuracy at a relatively modest computational cost.

The ability of IPRO to predict non-native ligand binding propensity was also tested using the Barnase protein as a DM. Barnase was redesigned to improve binding to pre-mRNA-splicing factor CWC26 (PDB entry 2MKC).[64] This ligand was selected because it was a short peptide that did not necessitate additional parameterization. The splicing factor was manually docked nearby the Barnase surface using PyMOL.[65] Any Barnase residue containing an atom within 4.5 Å of the splicing factor was included as a DP. IPRO was used using default values, and the interaction energy was improved from 42.02 kcal mol^{-1} to -370.75 kcal mol^{-1} by incorporating the following mutations: E29Y, A32I, G34A, G68A, and R69M. The initial and final structures are shown in Figure 9. This mutant not only eliminates the severe steric repulsion from the initial structure but also creates several favorable Lennard–Jones interactions. A32I in Barnase creates an attractive dispersion force with W332 of the splicing factor, and G68A of Barnase interacts favorably with P305. E29Y, G34A, and R69M help stabilize the Barnase conformation created by mutations A32I and G68A.

Conclusions

The IPRO suite of programs was designed as a unified portal with shared source code for integrating many computational protein engineering methods previously developed. All of the programs described here are run through the same user interface, the "Start_Experiment.py" program. This program prompts the user for all of the information that is required to run an experiment and validates that the provided information is acceptable. The user must answer "yes" and "no" questions, unless more detailed specifications are mandated. Once the specifications are provided, the program creates the experiment and identifies any problems.

All programs in the IPRO Suite now share the same code foundation so that there is a unique interface and ability to combine different methods and tasks. The code is also written in a modular, object-oriented fashion so that it is easy to add additional features to the IPRO suite of programs without causing compatibility issues. In this way, we intend to add new features to the IPRO suite of programs (e.g., supporting new force fields) as well as adding new computational protein engineering methods to this common suite of programs as they are developed.

The various programs in the IPRO suite have been successfully used for a variety of protein engineering applications in the last decade. Now they have been integrated together, improved for faster calculations (e.g., OptGraft), and modified to give improved confidence in the results (e.g., ensemble structure refinements). There is a new, single user interface that is intended to be easy to use for someone without extensive programming or protein engineering experience, opening up the IPRO suite to novice users. This simplified user interface was used to demonstrate the capabilities of both Mutator and IPRO using largely the same set of inputs. Mutator was successfully applied to correlate experimental binding free energies with interaction energies, and IPRO was used to improve binding for a non-native ligand. The input and output files for both of these demonstrations can be found at http://maranas.che.psu.edu.

Acknowledgment

The authors would like to thank Ariel Langevin for testing various inputs to the IPRO suite of programs.

Keywords: IPRO · proteins · enzymes · antibodies · computational design

How to cite this article: R. J. Pantazes, M. J. Grisewood, T. Li, N. P. Gifford, C. D. Maranas *J. Comput. Chem.* **2015**, *36*, 251–263. DOI: 10.1002/jcc.23796

[1] V. Menon, M. Rao, *Prog. Energy Combust. Sci* **2012**, *38*, 522.
[2] C. Xu, Y. Qin, Y. Li, Y. Ji, J. Huang, H. Song, J. Xu, *Bioresour. Technol.* **2010**, *101*, 9560.
[3] O. Leavy *Nat. Rev. Immunol.* **2010**, *10*, 297.
[4] J. A. Brannigan, A. J. Wilkinson, *Nat. Rev. Mol. Cell Biol.* **2002**, *3*, 964.
[5] R. J. Pantazes, M. J. Grisewood, C. D. Maranas, *Curr. Opin. Struct. Biol.* **2011**, *21*, 467.
[6] M. M. Meyer, L. Hochrein, F. H. Arnold, *Protein Eng. Des. Sel.* **2006**, *19*, 563.
[7] G. L. Moore, C. D. Maranas, *Proc. Natl. Acad. Sci. USA* **2003**, *100*, 5091.
[8] G. L. Moore, C. D. Maranas, *AIChE J.* **2004**, *50*, 262.
[9] C. R. Otey, M. Landwehr, J. B. Endelman, K. Hiraga, J. D. Bloom, F. H. Arnold, *PLoS Biol*. **2006**, *4*, e112.
[10] R. J. Pantazes, M. C. Saraf, C. D. Maranas, *Protein Eng. Des. Sel.* **2007**, *20*, 361.
[11] M. C. Saraf, C. D. Maranas, *Protein Eng.* **2003**, *16*, 1025.
[12] C. A. Voigt, C. Martinez, Z. G. Wang, S. L. Mayo, F. H. Arnold, *Nat. Struct. Biol.* **2002**, *9*, 553.
[13] S. Bjelic, L. G. Nivon, Celebi-N. Olcum, G. Kiss, C. F. Rosewall, H. M. Lovick, E. L. Ingalls, J. L. Gallaher, J. Seetharaman, S. Lew, G. T. Montelione, J. F. Hunt, F. E. Michael, K. N. Houk, D. Baker, *ACS Chem. Biol.* **2013**, *8*, 749.
[14] D. N. Bolon, S. L. Mayo, *Proc. Natl. Acad. Sci. USA* **2001**, *98*, 14274.
[15] R. A. Friesner, V. Guallar, *Ann. Rev. Phys. Chem.* **2005**, *56*, 389.
[16] M. P. Frushicheva, J. Cao, Z. T. Chu, A. Warshel, *Proc. Natl. Acad. Sci. USA* **2010**, *107*, 16869.
[17] L. Jiang, E. A. Althoff, F. R. Clemente, L. Doyle, D. Rothlisberger, A. Zanghellini, J. L. Gallaher, J. L. Betker, F. Tanaka, C. F. Barbas, 3rd, D. Hilvert, K. N. Houk, B. L. Stoddard, D. Baker, *Science* **2008**, *319*, 1387.
[18] S. D. Khare, Y. Kipnis, P. Greisen, Jr., R. Takeuchi, Y. Ashani, M. Goldsmith, Y. Song, J. L. Gallaher, I. Silman, H. Leader, J. L. Sussman, B. L. Stoddard, D. S. Tawfik, D. Baker, *Nat. Chem. Biol.* **2012**, *8*, 294.
[19] P. M. Murphy, J. M. Bolduc, J. L. Gallaher, B. L. Stoddard, D. Baker, *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 9215.
[20] H. K. Privett, G. Kiss, T. M. Lee, R. Blomberg, R. A. Chica, L. M. Thomas, D. Hilvert, K. N. Houk, S. L. Mayo, *Proc. Natl. Acad. Sci. USA* **2012**, *109*, 3790.
[21] F. Richter, R. Blomberg, S. D. Khare, G. Kiss, A. P. Kuzin, A. J. Smith, J. Gallaher, Z. Pianowski, R. C. Helgeson, A. Grjasnow, R. Xiao, J. Seetharaman, M. Su, S. Vorobiev, S. Lew, F. Forouhar, G. J. Kornhaber, J. F. Hunt, G. T. Montelione, L. Tong, K. N. Houk, D. Hilvert, D. Baker, *J. Am. Chem. Soc.* **2012**, *134*, 16197.
[22] W. Yang, Y. Pan, F. Zheng, H. Cho, H. H. Tai, C. G. Zhan, *Biophys. J.* **2009**, *96*, 1931.
[23] A. Zanghellini, L. Jiang, A. M. Wollacott, G. Cheng, J. Meiler, E. A. Althoff, D. Rothlisberger, D. Baker, *Protein Sci.* **2006**, *15*, 2785.
[24] P. Gainza, K. E. Roberts, I. Georgiev, R. H. Lilien, D. A. Keedy, C. Y. Chen, F. Reza, A. C. Anderson, D. C. Richardson, J. S. Richardson, B. R. Donald, *Methods Enzymol.* **2013**, *523*, 87.
[25] F. E. Boas, P. B. Harbury, *Curr. Opin. Struct. Biol.* **2007**, *17*, 199.
[26] J. Ashworth, J. J. Havranek, C. M. Duarte, D. Sussman, R. J. Monnat, Jr., B. L. Stoddard, D. Baker, *Nature* **2006**, *441*, 656.
[27] T. Kortemme, D. Baker, *Curr. Opin. Chem. Biol.* **2004**, *8*, 91.
[28] I. Georgiev, in Retrovirology, AIDS Vaccine 2012, Boston, MA, USA, Sept 9–12, **2012**; Springer: New York, 2012.
[29] B. I. Dahiyat, S. L. Mayo, *Science* **1997**, *278*, 82.
[30] B. Kuhlman, G. Dantas, G. C. Ireton, G. Varani, B. L. Stoddard, D. Baker, *Science* **2003**, *302*, 1364.
[31] D. Rothlisberger, O. Khersonsky, A. M. Wollacott, L. Jiang, J. De Chancie, J. Betker, J. L. Gallaher, E. A. Althoff, A. Zanghellini, O. Dym, S. Albeck, K. N. Houk, D. S. Tawfik, D. Baker, *Nature* **2008**, *453*, 190.
[32] C. Y. Chen, I. Georgiev, A. C. Anderson, B. R. Donald, *Proc. Natl. Acad. Sci. USA* **2009**, *106*, 3764.
[33] B. R. Donald, in Algorithms in Structural Molecular Biology; The MIT Press: Cambridge, MA, **2011**; Vol. 1, Chapter 11, pp 87–96.
[34] D. Baker, *Protein Sci.* **2010**, *19*, 1817.
[35] S. M. Lippow, B. Tidor, *Curr. Opin. Biotech.* **2007**, *18*, 305.
[36] H. Fazelinia, P. C. Cirino, C. D. Maranas, *Biophys. J.* **2007**, *92*, 2120.
[37] M. C. Saraf, G. L. Moore, N. M. Goodey, V. Y. Cao, S. J. Benkovic, C. D. Maranas, *Biophys. J.* **2006**, *90*, 4167.
[38] G. A. Khoury, H. Fazelinia, J. W. Chin, R. J. Pantazes, P. C. Cirino, C. D. Maranas, *Protein Sci.* **2009**, *18*, 2125.
[39] H. Fazelinia, P. C. Cirino, C. D. Maranas, *Protein Sci.* **2009**, *18*, 180.
[40] M. J. Grisewood, N. P. Gifford, R. J. Pantazes, Y. Li, P. C. Cirino, M. J. Janik, Maranas, C. D. *PLoS One* **2013**, *8*, e75358.
[41] R. J. Pantazes, C. D. Maranas, *Protein Eng. Des. Sel.* **2010**, *23*, 849.
[42] R. J. Pantazes, Maranas, C. D. *BMC Bioinf.* **2013**, *14*, 168.
[43] B. R. Brooks, C. L. Brooks, 3rd, A. D. Mackerell, Jr., L. Nilsson, R. J. Petrella, B. Roux, Y. Won, G. Archontis, C. Bartels, S. Boresch, A. Caflisch, L. Caves, Q. Cui, A. R. Dinner, M. Feig, S. Fischer, J. Gao, M. Hodoscek, W. Im, K. Kuczera, T. Lazaridis, J. Ma, V. Ovchinnikov, E. Paci, R. W. Pastor, C. B. Post, J. Z. Pu, M. Schaefer, B. Tidor, R. M. Venable, H. L. Woodcock, X. Wu, W. Yang, D. M. York, M. J. Karplus, *Comput. Chem.* **2009**, *30*, 1545.
[44] H. M. Berman, J. Westbrook, Z. Feng, G. Gilliland, T. N. Bhat, H. Weissig, I. N. Shindyalov, P. E. Bourne, *Nucleic Acids Res* **2000**, *28*, 235.
[45] D. T. Dransfield, E. H. Cohen, Q. Chang, L. G. Sparrow, J. D. Bentley, O. Dolezal, X. Xiao, T. S. Peat, J. Newman, P. A. Pilling, T. Phan, I. Priebe, G. V. Brierley, N. Kastrapeli, K. Kopacz, D. Martik, D. Wassaf, D. Rank, G. Conley, Y. Huang, T. E. Adams, L. Cosgrove, *Mol Cancer Ther* **2010**, *9*, 1809.
[46] F. F. Vajdos, M. Ultsch, M. L. Schaffer, K. D. Deshayes, J. Liu, N. J. Skelton, A. M. de Vos, *Biochemistry* **2001**, *40*, 11022.
[47] M. S. Lee, M. Feig, F. R. Salsbury, Jr., C. L. Brooks, 3rd. *J. Comput. Chem.* **2003**, *24*, 1348.
[48] T. Lazaridis, M. Karplus, *Proteins* **1999**, *35*, 133.
[49] R. L. Dunbrack, F. E. Cohen, *Protein Sci.* **1997**, *6*, 1661.
[50] B. R. Brooks, R. E. Bruccoleri, B. D. Olafson, D. J. States, S. Swaminathan, M. Karplus, *J. Comput. Chem.* **1983**, *4*, 187.

[51] T. Keitel, A. Kramer, H. Wessner, C. Scholz, Schneider-J. Mergener, W. Hohne, *Cell* **1997**, *91*, 811.

[52] S. C. Lovell, I. W. Davis, Arendall, W. B., 3rd; de P. I. Bakker, J. M. Word, M. G. Prisant, J. S. Richardson, D. C. Richardson, *Proteins* **2003**, *50*, 437.

[53] S. Ventura, M. C. Vega, E. Lacroix, I. Angrand, L. Spagnolo, L. Serrano, *Nat. Struct. Biol.* **2002**, *9*, 485.

[54] ILOG. ILOG CPLEX 11.0 User's Manual; ILOG Inc.: Mountain View, CA, **2007**.

[55] GAMS. **2013**, GAMS Development Corporation, GAMS, www.gams.com/index.htm (accessed Nov 24, 2014).

[56] F. Ehrenmann, V. Giudicelli, P. Duroux, M. P. Lefranc, *Cold Spring Harb. Protoc.* **2011**, *2011*, 726.

[57] M. P. Lefranc, *Cold Spring Harb. Protoc.* **2011**, *2011*, 633.

[58] M. P. Lefranc, *Cold Spring Harb. Protoc.* **2011**, *2011*, 643.

[59] M. P. Lefranc, C. Pommie, Q. Kaas, E. Duprat, N. Bosc, D. Guiraudou, C. Jean, M. Ruiz, Da I. Piedade, M. Rouard, E. Foulquier, V. Thouvenin, G. Lefranc, *Dev Comp Immunol* **2005**, *29*, 185.

[60] M. P. Lefranc, V. Giudicelli, C. Ginestoux, Jabado-J. Michaloud, G. Folch, F. Bellahcene, Y. Wu, E. Gemrot, X. Brochet, J. Lane, L. Regnier, F. Ehrenmann, G. Lefranc, P. Duroux, *Nucleic Acids Res.* **2009**, *37*, D1006.

[61] A. M. Buckle, G. Schreiber, A. R. Fersht, *Biochemistry* **1994**, *33*, 8878.

[62] I. H. Moal, J. Fernández-Recio, *Bioinformatics* **2012**, *28*, 2600.

[63] V. Potapov, M. Cohen, G. Schreiber, *Protein Eng. Des. Sel.* **2009**, *22*, 553.

[64] P. Wysoczanski, C. Schneider, S. Xiang, F. Munari, S. Trowitzsch, M. C. Wahl, R. Luhrmann, S. Becker, M. Zweckstetter, *Nat. Struct. Mol. Biol.* **2015**, *21*, 911–918.

[65] Schrodinger, LLC, PyMOL, **2010**, www.pymol.org (accessed Nov 24, 2014).

Received: 30 September 2014
Revised: 30 October 2014
Accepted: 8 November 2014
Published online on 2 December 2014